Shadows for Cel Animation

Lena Petrović * Brian Fujito * Lance Williams ° Adam Finkelstein *

* Princeton University
° Disney Feature Animation

Abstract

We present a semi-automatic method for creating shadow mattes in cel animation. In conventional cel animation, shadows are drawn by hand, in order to provide visual cues about the spatial relationships and forms of characters in the scene. Our system creates shadow mattes based on hand-drawn characters, given high-level guidance from the user about depths of various objects. The method employs a scheme for "inflating" a 3D figure based on hand-drawn art. It provides simple tools for adjusting object depths, coupled with an intuitive interface by which the user specifies object shapes and relative positions in a scene. Our system obviates the tedium of drawing shadow mattes by hand, and provides control over complex shadows falling over interesting shapes.

Keywords: Shadows, cel animation, inflation, sketching, NPR.
URL: http://www.cs.princeton.edu/gfx/proj/cel_shadows

1 Introduction

Shadows provide important visual cues for depth, shape, contact, movement, and lighting in our perception of the world [5, 17]. In cel animation, a moving figure and background scenery are illustrated in different layers with different styles, and therefore shadows play an especially crucial role by integrating the character into the background. According to Thomas and Johnston, two of Disney's most renowned animators, shadows were used in cel animation even from the very early days "because they anchored the figure to the ground. Without some kind of contact with the background, the characters seemed to float around, walking on air, no matter how much weight had been animated into their movements." [12] Traditionally, shadow mattes have been drawn by hand, and while modern digital image manipulation tools provide simple operations that assist in the creation of mattes, the process is still largely manual. In this paper, we present a semi-automatic method of creating shadow mattes from the hand-drawn outlines of moving figures. The process requires relatively little effort per frame, and generates plausible shadows cast by complex shapes over interesting background scenery such as walls, stairs and statues.

In rare cases where the shadow itself becomes a focal point for the viewer's attention, the shadow mattes *should* be drawn by hand, because they embody an artistic interpretation of the scene. In our work, we are addressing the rest of the shadow mattes – shadows that serve to anchor the character to the ground, enhance the form of the figure, or suggest lighting or mood. These represent the majority of the shadow mattes in cel animation, and the work of creating them is considerable. Unfortunately, it is not yet possible to *fully* automate the process of creating shadow mattes based on the line art; understanding the shapes suggested by the line art is tantamount to solving the computer vision problem, and is subject to ambiguities in interpretation. Thus, our system requires a small amount of user input – less effort than would be required to draw the mattes by hand. Once the user has set up the scene, it is easy to alter the lighting conditions to produce very different kinds of shadows. The benefits of such a system are a reduction in effort, an increase in control and flexibility, and the ability to create plausible shadow mattes even for complex character-scene interactions.

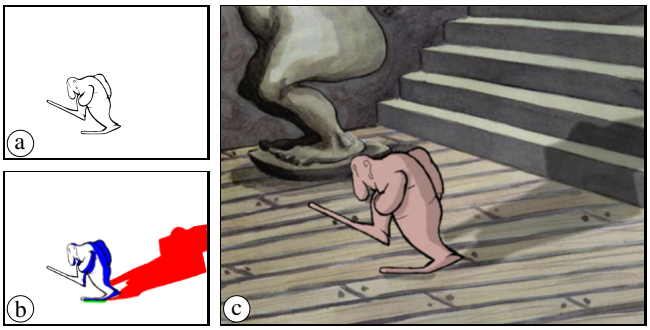

Figure 1: An example frame. (a) hand-drawn line art (b) shadow mattes created by our system (c) composited frame.

At a high level, our process works as follows. We begin with hand-drawn line art created by a traditional animator (Figure 1a), as well as hand-painted scenery created by a background artist. The user sketches over features in the painted background to establish the camera, ground plane, and background objects. Using character mattes integral to the compositing stage of the normal cel animation pipeline, we automatically "inflate" a 3D mesh for the character. The user specifies the depth for the character in the scene, as well as light positions. Next, based on the lights, the 3D character, and the background objects, the computer renders three types of shadow mattes for the character: *tone mattes* indicate both self-shadowing and shadows of other objects on the character (Figure 1b, blue); *contact shadow mattes* emphasize contact between the character and the ground (green); and *cast shadow mattes* specify shadows cast by the character onto the background scenery (red). Finally, we composite these mattes into the scene (Figure 1c) as part of the conventional cel animation pipeline.

The contributions of this work are: (1) application of "inflation" algorithms to create frame-by-frame 3D models from hand-drawn animation; (2) tools that allow the user to manipulate the inflated models of the artwork in three dimensions while preserving their image-space silhouettes; and (3) an intuitive user interface for describing shapes and relative 3D positions of both static and animated objects in the scene.

The remainder of this paper is organized as follows. In Section 2 we review related work. Section 3 describes the details of our process. In Section 4 we demonstrate the results of our working system. Section 5 concludes with observations and proposed areas of future work.

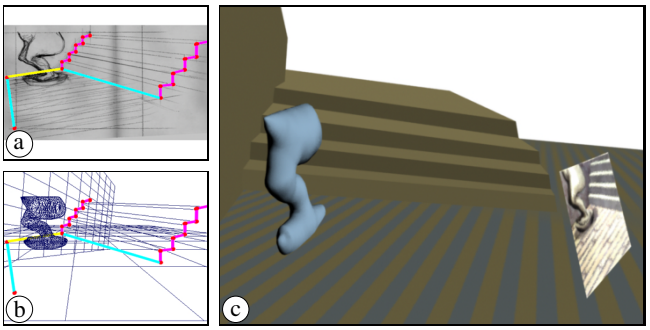

Figure 2: Specifying background. (a) marking features in artwork (b) 3D scene in wireframe (c) flat-shaded side-view of 3D scene.

2 Related Work

A variety of previous efforts have applied computer graphics in cel animation. Researchers have automated the image processing and compositing aspects of cel animation [2, 7, 11, 15], in which the shadow mattes created by our system could replace hand-drawn mattes. For the cel animation industry, 3D techniques are becoming increasingly prominent in production. For example, in Disney's *Tarzan* [16], jungle backgrounds were built and painted in 3D with "Deep Canvas," to give extra depth to the scene and allow camera fly-throughs [9]. For *The Prince of Egypt* [13], DreamWorks used "Exposure," allowing them to fly cameras through a 3D scene with sequential background shots specified by manipulating a series of painted 2D background cards [8]. 3D methods have also been used for cel animation in research contexts, for example in the design of *multiperspective panoramas* (static background paintings that are appropriate for a moving camera [22]) or *view-dependent geometry* (3D models that change shape depending on view direction [6]). Previously, we showed how to apply texture to a hand-animated character by warping a 3D model to match the line art [1]. While this method could be adapted for creating plausible shadow mattes, the process requires too much human effort for this application; it would be easier generally to draw the shadow mattes by hand.

Since shading and tones enhance our understanding of 3D shape, a number of cel animation projects have applied 3D computer graphics for shading. In the extreme, a character (for example, the giant in *The Iron Giant* [18]) is modeled and animated entirely in 3D and then rendered with a "cartoon shader." Most characters in cel animation are not designed in 3D, and therefore to invoke 3D shading techniques, one must form some kind of 3D representation of the 2D artwork. This project grew out of previous work [19, 20] where we used pixel-based inflation schemes to automatically build 3D representations from 2D art, and then rendered tones and shadows with strictly 2D interactions. In this paper, we employ a 3D inflation technique, and develop technology that allows us to interactively stage a scene with 3D shadow interactions.

We adapt the inflation algorithm originally proposed by van Overveld and Wyvill [14], as implemented in a simpler form in the "Teddy" system of Igarashi *et al.* [4]. We use Teddy because it generates 3D forms that yield plausible tones and cast shadows. However, we modify their method to account for a perspective camera, requiring that the figure aligns with the artwork on its silhouette, as seen from the camera. The more significant departure from the Teddy system is that we build the character up in multiple layers, rather than extruding limbs from the main body. In our application, there are two advantages to the layer-based approach: it is more consistent with the cel animation pipeline, and it guarantees that the silhouette of each layer in the figure matches the line art.

Finally, our strategy for constructing the 3D scene is largely inspired by the "SKETCH" system of Zeleznik *et al.* [23].

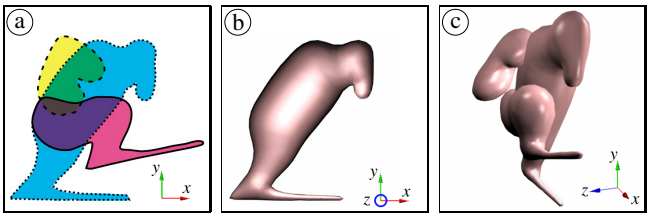

Figure 3: Building a three-layered character. (a) character mattes (b) middle-layer inflation (c) 3D character seen from side view.

3 The Process

Here we describe the process of creating shadow mattes based on hand-drawn art. Section 3.1 addresses construction of background scenery based on a few simple gestures by the user. Section 3.2 describes inflation of 3D characters from line art, and Section 3.3 presents tools for adjusting the relative depths of the 3D figures in the scene. Finally, Section 3.4 describes setting up lights, rendering shadow mattes, and compositing them into the artwork.

3.1 Constructing Background Scenery

The first stage of the process of creating the scene is to construct a background. In order to establish the relationship between the camera and the scene, we begin with several assumptions: a fixed field of view and aspect ratio for the camera (in our tests, 83° wide with an aspect of 4:3), and known camera roll and ground plane tilt (both are upright). These assumptions work for a broad class of scenes, and may be easily modified by the user to work for other scenes. Next, in order to establish the pitch of the camera relative to the ground plane, the user sketches over the background art a pair of parallel lines in the ground plane, for example the cyan lines in Figure 2a. In a perspective image, all parallel lines on the ground plane that intersect in the image plane will intersect on the horizon. Thus, even if the horizon is not visible in the scene we can find it by intersecting the parallel lines given by the user. The height h of the horizon relative to the center of the image determines[1] the pitch ϕ of the camera by the simple relationship:

$$\phi = \arctan(h/d)$$

where d is the distance from the camera to the image plane (which we set arbitrarily to 1). The only remaining camera parameters are its yaw and its height above the ground plane. Since at this stage there are no objects in the scene other than the ground plane, the yaw is arbitrary and we set it to zero. The camera height establishes a scale for the scene that is also arbitrary. For example, in Figure 1 we see a man in a room, but whether this room is the size of a bread box or a warehouse has no impact on the size or shape of his shadow projected onto the image plane. Thus, we arbitrarily choose the height of the camera, only taking care that the entire image plane is guaranteed to be above the ground plane. Now we establish a coordinate system for the scene: we take the origin to be the center of the image plane, the x and y axes to be, respectively, the horizontal and vertical axes of the image, and the -z axis to be the camera look direction.

Taking inspiration from the SKETCH system [23], we construct objects in the scene relative to the ground plane using simple gestures. Walls are built perpendicular to the ground by specifying the line of intersection with the ground plane. The user can create more complicated objects consisting of multiple polygonal faces

[1]This method fails for cameras that are looking straight up or straight down, and in these cases we would ask the user to set the pitch by hand.

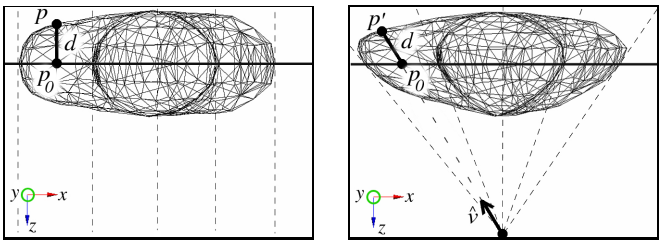

Figure 4: Inflating a 3D figure from 2D line art using perspective. Left: orthographic extrusion. Right: perspective extrusion.

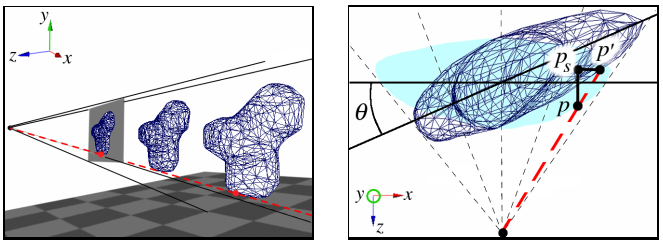

Figure 5: Adjusting depth. Left: ball grows as it moves back to contact point. Right: Depth-shear preserves perspective outline.

(such as stairs and boxes) by specifying two polylines on the object, starting with ground contact points. The object is constructed so that it conforms to the sketched lines and its neighboring polygonal faces are perpendicular. Figure 2 shows the user-identified features in the image: cyan for the ground lines, yellow for the wall line, and violet for the stairs. Smooth, organic 3D background objects, such as the statue in Figure 2, are built using the character inflation and placement methods described in Sections 3.2 and 3.3.

3.2 Inflating 3D Models

Having specified the background, the next step is to inflate the line art to form meshes representing the character in 3D. Our goal is to create 3D figures that cast plausible shadows under various lighting conditions. First, we convert the line art into *character mattes* – bitmaps that define regions of the image covered by the character. This process is performed as a normal aspect of the cel animation pipeline, because these mattes are used for filling and clipping when the character is composited into the scene. For further control in our system, we often divide the character mattes into multiple layers, so that each may be placed at a different depth. These separate layers help to cast plausible shadows under lighting conditions that reveal the geometry of the character. For the character in Figure 3, the arm (inflated from the yellow matte) could cast a shadow from an overhead light onto the leg (magenta). Layer mattes may need to be extended past visible boundaries to cast reasonable shadows. For example, a matte for the man's right arm in Figure 1 would extend into his body, with the body-arm boundary being preserved by placing the arm behind the body (as described in Section 3.3). Converting line art into character mattes must be a manual process, because it requires a visual interpretation of the scene. However, it is fairly easy to perform using digital image editing tools such as Adobe Photoshop. In some cases, these layer mattes may already be available to us, as animation houses may split characters into multiple layers for separate animation.[2]

Next we "inflate" each layer in the character matte to form rotund 3D shapes; for example the shape shown in Figure 3b is inflated from the cyan region in Figure 3a. Our inflation scheme is based on the Teddy system of Igarashi *et al.* [4], which finds the chordal axis of a closed curve, lifts it out of the plane, and lofts a surface between the curve and its axis. Many other inflation methods might be used at this stage. We chose Teddy because it is simple and fast, and produces smooth, bulky shapes that yield reasonable cast shadows and tone mattes. As input to Igarashi's algorithm, each layer in the character mattes should be described by a simple, closed, 2D polyline, i.e. there should be no holes. We use an automatic tracing tool to convert the matte to a closed curve, and then resample it as a polyline. Teddy then elevates a surface whose intersection with the image plane is the input polyline.

[2]This is more common in television animation where individual layers tend to be reused, but less common in feature animation.

If we were using an orthographic camera, we would be done. However, with a perspective camera, this scheme will not produce a 3D shape whose silhouette is exactly aligned with the original matte. Therefore, we adjust the resulting shape as follows. Every point p on the lofted surface is a signed distance d from its corresponding point p_0 in the image plane of the orthographic camera (Figure 4). For the perspective camera, we find the point p' on the ray from the camera to p_0:

$$p' = p_0 + d\hat{v}$$

where $\hat{v}$ is the normalized vector from the camera to p_0. By sending every point p in the Teddy-lofted surface through this transform, the silhouette of the resulting shape conforms exactly to the matte.

3.3 Specifying Depth

Now that we have inflated the layers of the character from the image plane, we need to give them depth in the 3D world while preserving their image plane projection. We provide two projection-preserving depth adjustments: *depth-translation* and *depth-shear*. While one could implement more sophisticated projection-preserving warps, we designed these methods to minimize user interaction.

Depth-translation moves the figure out of the image plane to varying depths in the scene; we maintain its image plane projection using a uniform scale about the camera center. The user specifies the translation in one of two ways. First, she can mark a contact point in the image plane, and the system pushes the object back into the scene until the marked point reaches the ground plane, as shown in Figure 5a. Second, the user can push the object towards or away from the camera via a manipulator, watching the object move in relation to other objects in the scene as well as an approximate shadow cast by an overhead light onto the ground plane.

We also allow the user to control the relative depth across a single object via a projection-preserving *depth-shear*, which provides fine control over how shadows will be cast by this figure. For example, in Figure 3c, the arm has been sheared so that the hand is closer to the image plane (to the left in this pose) than the rest of the arm. Nonetheless, the image-plane projection of this object must remain unchanged. The user specifies the shear via a manipulator that provides an axis and an angle θ. Our algorithm works as follows. For every point p on the unsheared object (light blue in Figure 5b), we calculate p_s, the point that would result from sending p through the conventional shear by θ. We take our final point p' to be the point on the ray extending from the camera through p that has the same z value (depth) as p_s.

Depth adjustment is generally specified separately for all layers. However, for any layer this information may be keyframed across time. For example, by specifying a contact point in two frames of the animation of the ball shown in Figure 7 we are able to implicitly set depths for the ball in all of the other frames. Likewise, the contact points for the stomping man are keyframed, even though his body remains at approximately constant depth.

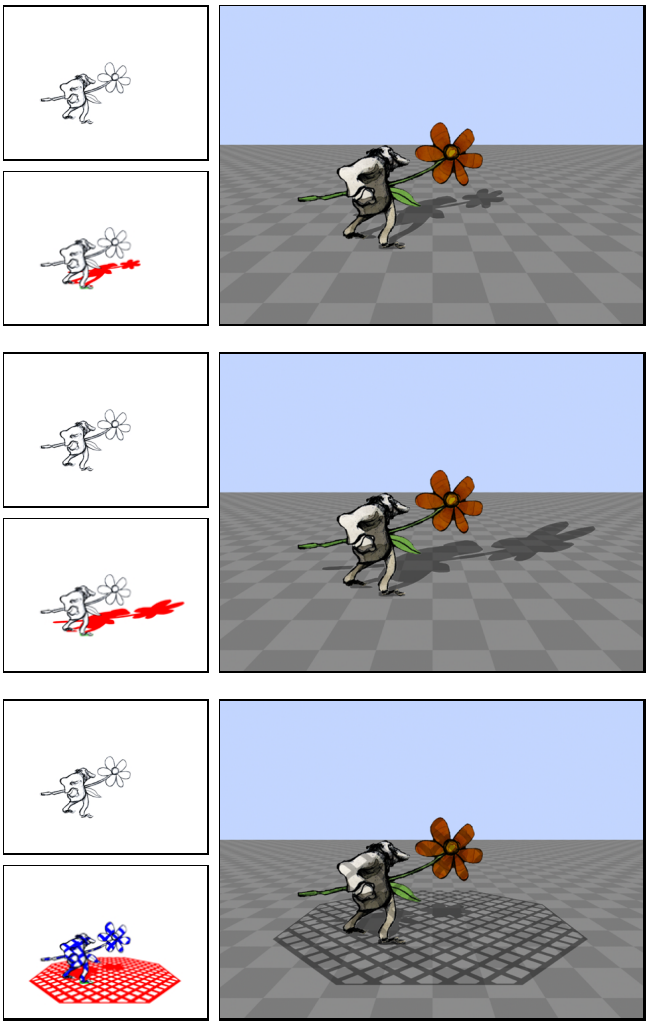

Figure 6: Specifying lights. From top to bottom: directional light, point light, gazebo gobo.

3.4 "Lights ... Camera ... Action!"

Now that we have constructed a background set and positioned a 3D character in the scene, setting up lights and rendering shadow mattes is straightforward. As shown in Figure 6, we obtain different shadow effects by using directional lights, point lights, or by using a "gobo."[3] Within a range of lighting subjectively consistent with the painted background, lights may even be animated to provide fine control over where a shadow falls in the scene. When creating the sequence shown in Figure 8, we animated the light position to cause the shadow to fall on the legs of the statue early in the clip and extend up the stairs later.

To render the mattes for tones and cast shadows, we use a standard ray tracer with custom shaders. For example, for on-character tone mattes we have implemented a cartoon shader that thresholds all colors to either white or black based on a diffuse lighting calculation. Due to sampling and approximation errors in 3D mesh construction, it is possible that the 3D figure used to render tone mattes does not exactly align with the line art, causing a small gap between the tone matte and the line art. In these cases,

[3] A "gobo", referred to in live action film as a "cuckaloris" or "cookie", is a device for projecting a shadow pattern (such the foliage in Figure 9, or in this case a gazebo) onto a scene.

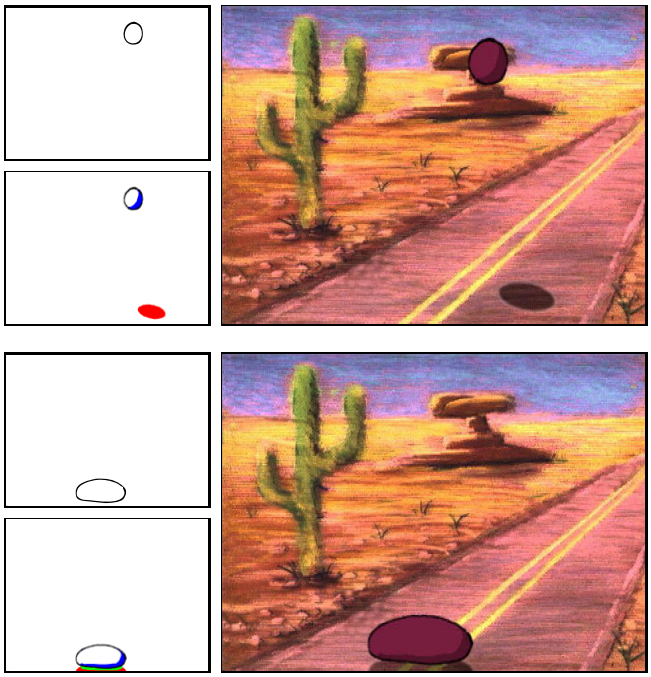

Figure 7: Bouncing ball casts shadow in desert scene.

we apply morphological dilation to the tone matte, and then clip it to the original character matte. To accommodate this operation, we actually render a separate tone matte for each layer in the figure and perform dilation and clipping on each independently before combining them.

We render contact shadows in two passes. First, we place an orthographic camera in the ground plane, looking up at the character, and use a far clipping plane that is just above the ground. With this camera, we capture an image of parts of the character that nearly touch the ground. Second, we re-project the image onto the ground plane, and view it from the regular camera for the scene, giving us contact shadows. Final dilation of the resulting matte ensures that it emerges from beneath the character.

Now we are ready to composite the frame. Tone mattes modulate the character's color, while mattes for cast shadows and contact shadows darken the background painting. In some cases, we blur the mattes to suggest softer shadows. Finally, we composite the shadowed character over the shadowed background, and then add the line art.

4 Experimental Results

In this section, we describe the animations that we created using our system, as well as the time and effort for building these sequences.[4] The animations are shown on the video proceedings.

Figure 7 shows two frames from a 33-frame sequence of a bouncing ball. We built a ground plane (from the lines of the road) and inflated each ball mesh as a single, animated layer. Since parts of the ball go off-frame in the last five frames, we completed the character mattes (extending out of the frame) to get reasonable shadows. We specified the depth of the bouncing ball using ball-to-ground contact points in two frames, and interpolated and extrapolated depths for all other frames.

[4] Our system is implemented on a 400 MHz Pentium II PC as a series of plug-ins for Alias|Wavefront Maya. For matte editing and compositing we use Adobe Photoshop, Softimage Eddie and NothingReal Shake.

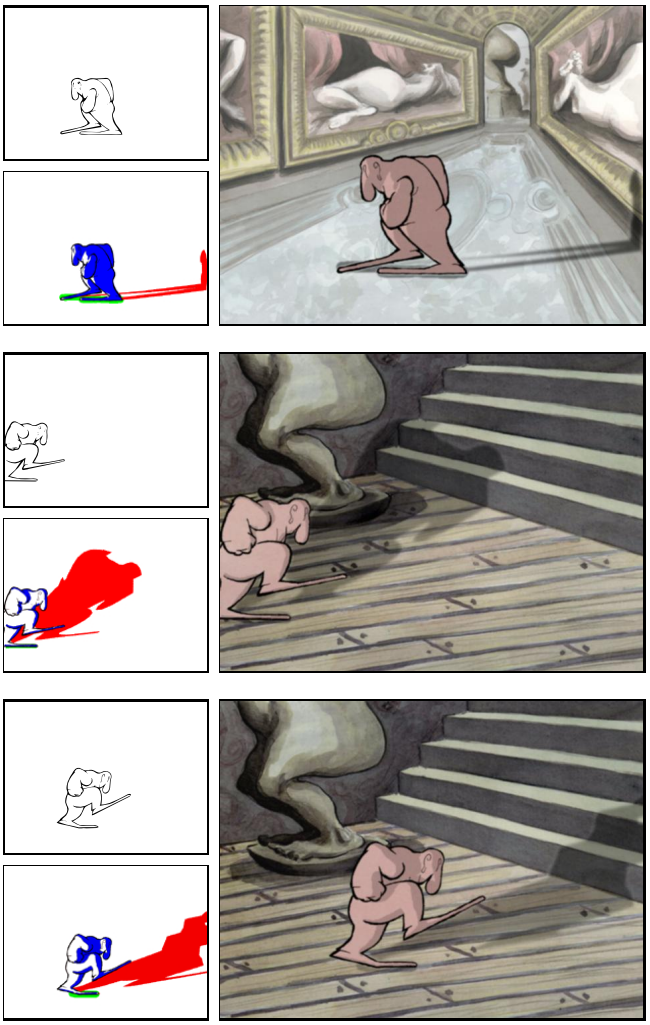

Figure 8: Stomping man in action, with two different backgrounds.

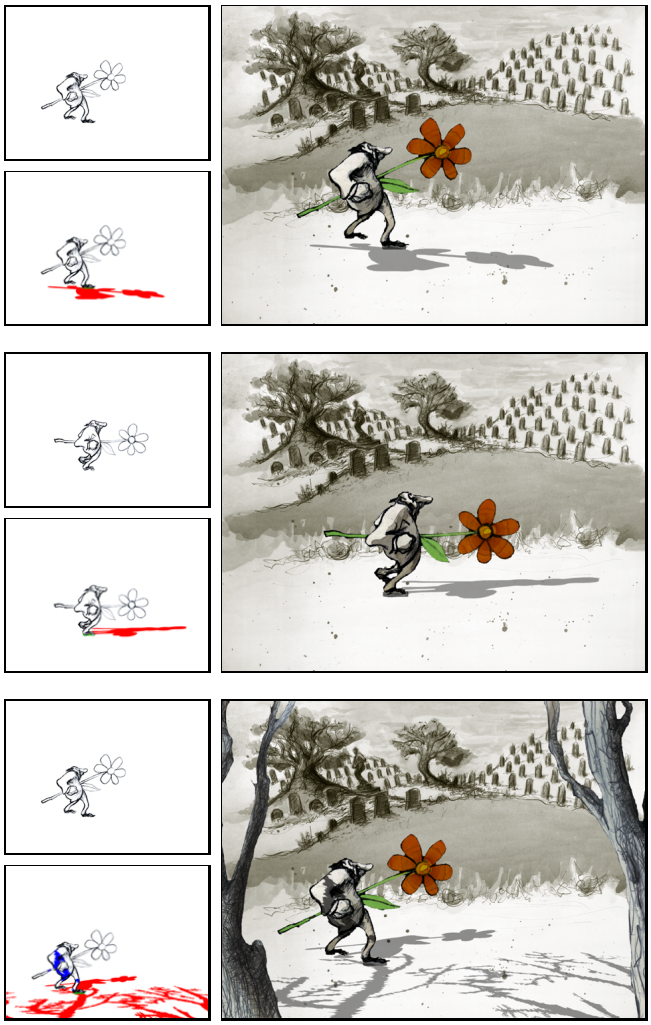

Figure 9: Man with flower. The bottom image uses a tree gobo.

Figure 8 shows three frames of a stomping man – one in a hallway and two near some stairs. For both scenes, we built the stomping man from a 16-frame, hand-drawn walk cycle. For each of the 16 frames, we split up the character into three or four layers (depending on the visibility of the far arm) and specified different relative depths and shears in each layer. Our system then built offset copies of these meshes to form 10 stomp cycles (160 frames). In the hallway scene, we placed a near-horizontal directional light that casts a long shadow breaking up the far wall. In the staircase scene, we subtly animated a point light, as described in Section 3.4.

In Figure 9 we show three test frames from a work in progress of an old man carring a flower. The upper two frames illustrate varying the light direction. Because of the style of the artwork, these frames do not use tone mattes. The bottom frame shows the body of the man and the ground receiving a shadow from a tree gobo.

The bulk of the human effort involved in our system consists of the following: (1) specifying the background – 1 or 2 minutes, even for complex scenes such as the stair scene; (2) creating character mattes – 2 minutes per layer per frame;[5] (3) specifying depth information for each layer – roughly 30 seconds per layer per frame;[6] (4) specifying lights – under 1 minute. The human effort required to specify layer mattes and depth information may be substantial. However, several factors mitigate this cost. First, layer segmentation is only required insofar as the lighting and geometry demand it. Second, the task of layering and depth specification requires minimal artistic interpretation, and might be relegated to junior staff. Third, specifying depth using our interface is simple, so that even if the layers frequently change depth across frames, the work of adjusting layers is faster, simpler, and more easily adapted than drawing shadows by hand. Finally, once the scene has been established, changing the lighting is easy. Thus, in general, our system requires relatively little human effort, and allows users to cast interesting shadows that would otherwise be quite tedious to draw by hand.

The most computationally expensive aspect of our system is ray tracing shadow mattes, each of which takes roughly 45 seconds at 640×480.[7] This time is characteristic for ray-traced shadows; other rendering regimes (e.g. depth buffer shadows or shadows cast by area lights) could be expected to vary in computational expense.

[5] For the stomping man, it took 2 hours to build mattes for a 16-frame cycle with 4 layers. For the 9 other cycles, character mattes were simply offset with no human effort.

[6] For the entire ball sequence, the depths were specified in 1 minute using only two contact points. For the stomping man, setting depths took about 30 minutes for 16 frames; the remaining 144 frames were cycled.

[7] Computing shadows took less than 2 hours for the ball (33 frames, 3 mattes) and 12 hours for the stomping man (160 frames, 6 mattes).

5 Conclusion and Future Work

This paper presents a method for semi-automatic creation of shadow mattes for cel animation. The process reduces human effort normally required for painting shadow mattes. The system yields plausible shadows, even for complex characters and scenes. We show examples with dramatic lighting – directional lights and point lights casting very oblique shadows, and even gobos – in order to demonstrate the effectiveness of the method with such lights.

Setting up the scene and generating shadow mattes is relatively easy. Furthermore, once this task is accomplished it is trivial to subsequently change the lighting conditions and experiment with different effects. Thus, animators and directors can easily adjust the lighting (within a range consistent with the underlying painted artwork), in contrast to the traditional method wherein the shadow artist would redraw all of the mattes.

While the shadows in our examples are reasonable, there are some characters or background objects for which our inflation technique would be inappropriate. Either hand-drawn shadows or alternate modeling methods would address these problem cases.

This work suggests a number of areas for future investigation:

Applying computer vision techniques for understanding art. The most time-consuming aspect of our system is creation of layered character mattes. Applying computer vision techniques (e.g. [21, 24]) to automate layer specification would facilitate this process. Ultimately, tracking processes which adapt 3D models to drawn animation sequences [3] offer the potential of highly refined shadows and rendering effects.

Automatic light placement. Artists and directors accustomed to traditional hand-drawn techniques may be benefit from automatic light placement, based on either the background painting and its simple 3D model, or on crude hand-drawn samples (in the spirit of Schoeneman *et al.* [10]).

Shadow simplification. In traditional animation, hand-drawn shadows are often abstract rather than realistic. We would like to be able to simplify shadows, perhaps as a post-process in this system.

Acknowledgements

Many of the ideas in this paper were advanced at DreamWorks by Galen Gornowicz, Saty Raghavachary, and Gigi Yates. We are extremely grateful to Grady Klein and Rob Jensen for creating the hand-drawn characters and background scenery shown here. Thanks to Rick Szeliski and Mike Salisbury for helpful discussions about constructing the scene.

This work was supported by an NSF CAREER Award and an Alfred P. Sloan Fellowship.

References

[1] CORRÊA, W. T., JENSEN, R. J., THAYER, C. E., AND FINKELSTEIN, A. Texture mapping for cel animation. *Computer Graphics (Proceedings of SIGGRAPH 98)*, 435–446.

[2] FEKETE, J., BIZOUARN, E., COURNARIE, E., GALAS, T., AND TAILLEFER, F. TicTacToon: A paperless system for professional 2-D animation. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 79–90.

[3] GORNOWICZ, G., AND WILLIAMS, L. Snap to it! Automatic 3D object and silhouette registration. Sketches and Applications, SIGGRAPH 2000.

[4] IGARASHI, T., MATSUOKA, S., AND TANAKA, H. Teddy: A sketching interface for 3D freeform design. *Computer Graphics (Proceedings of SIGGRAPH 99)*, 409–416.

[5] KERSTEN, D., MAMASSIAN, P., AND KNILL, D. C. Moving cast shadows induce apparent motion in depth. *Perception 26*, 2 (1997), 171–192. Also see: http://vision.psych.umn.edu/www/kersten-lab/demos/shadows.html.

[6] RADEMACHER, P. View-dependent geometry. *Computer Graphics (Proceedings of SIGGRAPH 99)*, 439–446.

[7] ROBERTSON, B. Disney lets CAPS out of the bag. *Computer Graphics World* (July 1994), 58–64.

[8] ROBERTSON, B. Mixed media. *Computer Graphics World* (Dec. 1998), 32–35.

[9] ROBERTSON, B. Deep background. *Computer Graphics World* (July 1999), 50–51.

[10] SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J., AND GREENBERG, D. Painting with light. *Computer Graphics (Proceedings of SIGGRAPH 93)*, 143–146.

[11] SHANTZIS, M. A. A model for efficient and flexible image computing. *Computer Graphics (Proceedings of SIGGRAPH 94)*, 147–154.

[12] THOMAS, F., AND JOHNSTON, O. *Disney Animation: The Illusion of Life*. Walt Disney Productions, New York, 1981.

[13] UNIVERSAL STUDIOS / DREAMWORKS. *The Prince of Egypt*. Movie, 1999.

[14] VAN OVERVELD, K., AND WYVILL, B. Polygon inflation for animated models: A method for the extrusion of arbitrary polygon meshes. *Journal of Vision and Computer Animation 18* (1997), 3–16.

[15] WALLACE, B. A. Merging and transformation of raster images for cartoon animation. *Computer Graphics (Proceedings of SIGGRAPH 81)*, 253–262.

[16] WALT DISNEY PRODUCTIONS. *Tarzan*. Movie, 1999.

[17] WANGER, L., FERWERDA, J., AND GREENBERG, D. Perceiving spatial relationships in computer-generated images. *IEEE Computer Graphics and Applications, 12*, 3 (1992), 44–58.

[18] WARNER BROTHERS. *The Iron Giant*. Movie, 1999.

[19] WILLIAMS, L. 3D rendering effects for 2D animation. Sketches and Applications, SIGGRAPH 1999.

[20] WILLIAMS, L. Shading in two dimensions. *Proceedings of Graphics Interface 91*, 143–151.

[21] WILLIAMS, L. R. Topological reconstruction of a smooth manifold-solid from its occluding contour. Tech. Rep. 94-04, University of Massachusetts, Amherst, MA, 1994.

[22] WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. Multiperspective panoramas for cel animation. *Computer Graphics (Proceedings of SIGGRAPH 97)*, 243–250.

[23] ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. SKETCH: An interface for sketching 3D scenes. *Computer Graphics (Proceedings of SIGGRAPH 96)*, 163–170.

[24] ZHU, S., AND YUILLE, A. FORMS: A flexible object recognition and modelling system. *International Journal of Computer Vision 20*, 3 (1996), 187–212.